Assignment

Name – Yashica Garg
PRN – 20200802125
Subject – DNN

Problem Statement –
Creating a Multi-Layer ANN with TensorFlow

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning and developing neural networks faster and easier.

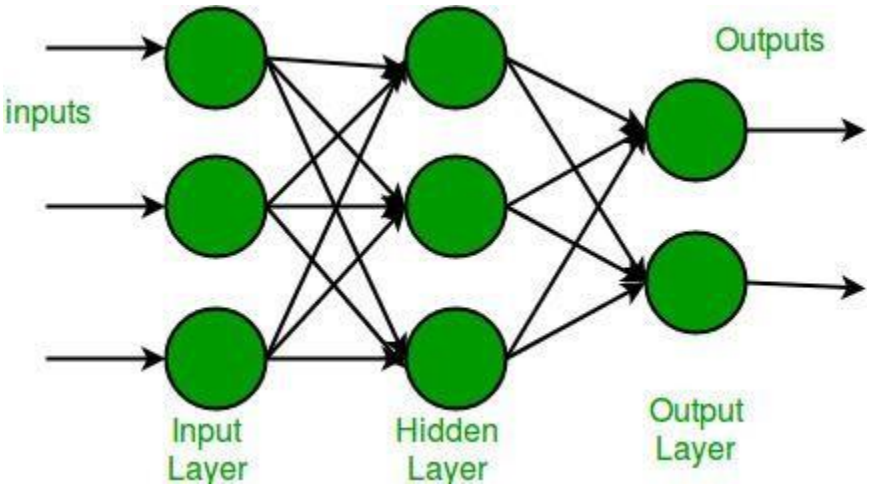
Machine learning is a complex discipline but implementing machine learning models is far less daunting than it used to be, thanks to machine learning frameworks—such as Google's TensorFlow—that ease the process of acquiring data, training models, serving predictions, and refining future results.

TensorFlow, which competes with frameworks such as PyTorch and Apache MXNet, can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation)-based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

Multi-layer Perceptron

Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.

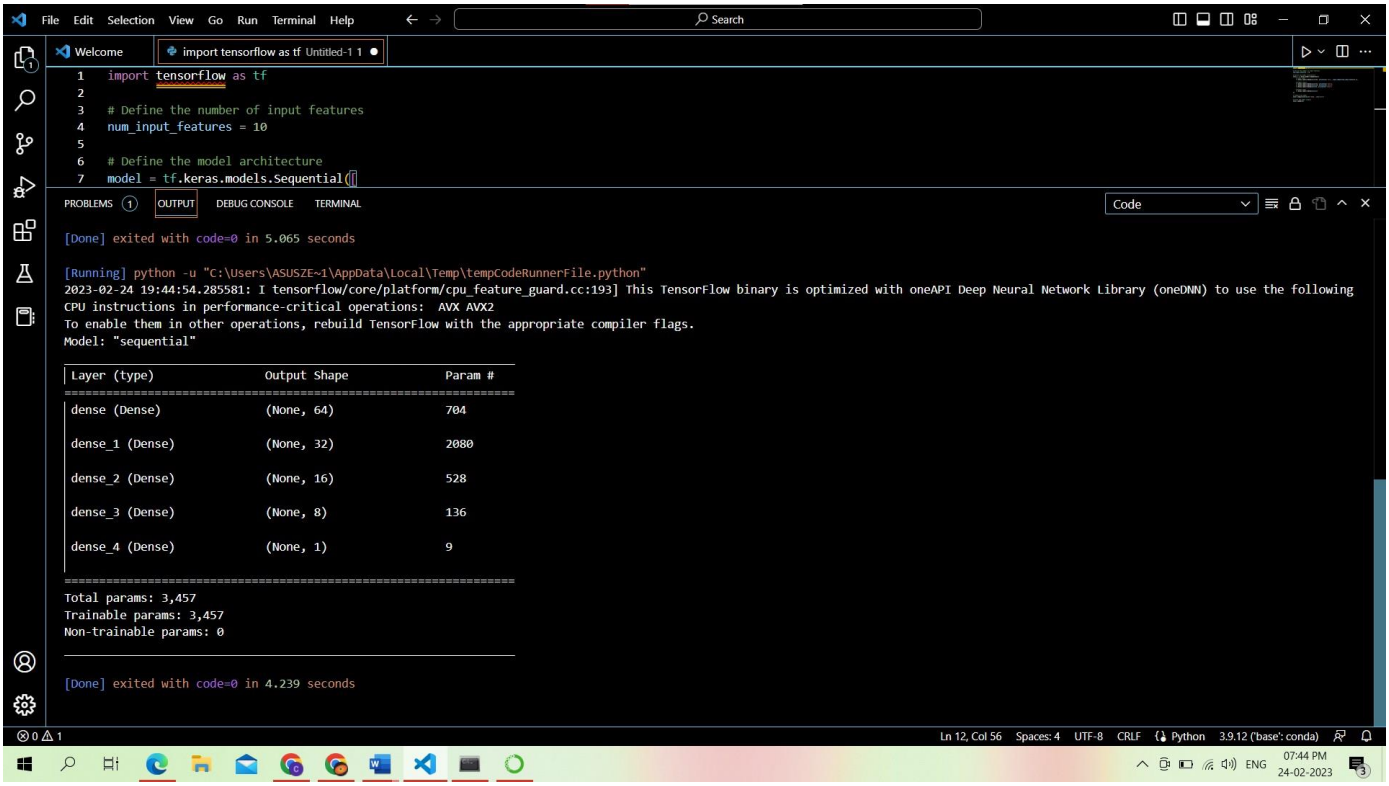
A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.



Code –

```
import tensorflow as tf

# Define the number of input features
num_input_features = 10

# Define the model architecture
model = tf.keras.models.Sequential([
    # Input layer
    tf.keras.layers.Dense(units=64, activation='relu', input_shape=(num_input_features,)),

    # Hidden layers
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=16, activation='relu'),
    tf.keras.layers.Dense(units=8, activation='relu'),

    # Output layer
    tf.keras.layers.Dense(units=1)
])

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Print the model summary
model.summary()
```

Output-


File Edit Selection View Go Run Terminal Help
Search
Welcome
import tensorflow as tf Untitled-1 1
1 import tensorflow as tf
2
3 # Define the number of input features
4 num_input_features = 10
5
6 # Define the model architecture
7 model = tf.keras.models.Sequential([
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL
Code
[Done] exited with code=0 in 5.065 seconds
[Running] python -u "C:\Users\ASUSZE~1\AppData\Local\Temp\tempCodeRunnerFile.python"
2023-02-24 19:44:54.285581: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model: "sequential"
Layer (type) Output Shape Param #
dense (Dense) (None, 64) 704
dense_1 (Dense) (None, 32) 2080
dense_2 (Dense) (None, 16) 528
dense_3 (Dense) (None, 8) 136
dense_4 (Dense) (None, 1) 9
Total params: 3,457
Trainable params: 3,457
Non-trainable params: 0
[Done] exited with code=0 in 4.239 seconds
Ln 12, Col 56 Spaces: 4 UTF-8 CRLF Python 3.9.12 ('base': conda)
ENG
07:44 PM
24-02-2023